REMARKS

In the Office Action, the Examiner rejected Claims 1-16, which were all of the then pending claims, over the prior art, principally U.s. Patent 6,247,172 (Dunn, et al.). The Examiner also objected to informalities in the specification and in Claim 5 and further rejected Claim 5 under 35 U.S.C. §112 as being indefinite.

With respect to the rejections of the claims over the prior art, Claims 1, 2, 4, 6, 7 and 10-16 were rejected under 35 U.S.C. §102 as being fully anticipated by Dunn, et al; and Claims 3, 5, 8 and 9 were rejected under 35 U.S.C. §103 as being obvious over Dunn, et al. in view of U.S. patent 6,412,109.

Independent Claims 1, 6, 7, 8, 10, 12, 13, 15 and 16 are being amended to better define the subject matters of these claims. Claims 3 and 5 are being amended to keep the language of these claims consistent with the language of Claim 1, and Claim 9 is being amended to keep the language of this claim consistent with the language of Claim 8. New Claims 17 and 18, which are dependent from Claim 1, are being added to describe preferred features of the invention.

Also, Claim 5 and the specification are being amended to address the Examiner's objections. In particular, in Claim 5, as the Examiner suggested, "said exception program" is being changed to "said exception process", and in the last line of the claim, "process was performed" is being changed to "process is performed". Further, on page 18 of the specification, the "Description of the Symbols" has been removed. It is noted that all of the reference numbers in this "Description" occur elsewhere in the specification, and it is not necessary to add any substitute for this "Description of the Symbols."

In view of these changes to the specification and to Claim 5, the Examiner is requested to reconsider and to withdraw the objections to the specification and to Claim 5 and the rejection of Claim 5 under 35 U.S.C. §112.

In addition, for the reasons discussed below, Claims 1-18 patentably distinguish over the prior art and are allowable. The Examiner is thus also asked to reconsider and to withdraw the rejection of Claims 1, 2, 4, 6, 7 and 10-16 under 35 U.S.C. §102 and the rejection of Claims 3, 5, 8 and 9 under 35 U.S.C. §103, and to allow Claims 1-18.

This invention relates to a computer for performing effective optimization for a program that includes a command that may cause an exception process. As explained in detail in the present application, for a program written in a programming language that establishes an exception handler for an exception process and shifts the processing to the exception handler, deterioration of optimization effects occurs if a command that may cause the execution of an exception process is included in a program.

The present invention effectively addresses this issue by providing an optimization execution unit for performing an optimization process for an object program written in a machine language, and a program modification unit for modifying that object program. This modifying is done in order to absorb a difference in content between the point of origin of an exception process, which occurs in response to the execution of a command in said object program, and an event handler whereat said exception process is performed.

Dunn fails to disclose or suggest the above-discussed program modification unit, and more specifically, the use of the event handler as described above. More specifically, in this invention, the compensation code "to absorb a difference..." has a different objective from Dunn's recovery block 42. The target machine state of compensation code of this invention is the "target machine" state at an exception handler. However, the target machine state of the recovery block of Dunn is the "legacy machine" state at the code that is throwing the current exception. Therefore, they are different locations and different states. In the approach disclosed in Dunn, the recovery block contains instructions that complete all functions necessary to restore the target machine state to "the legacy machine state" (Dunn column 3, line 13).

The legacy machine state is the machine state that could have resulted had the target platform executed "target code not optimized by the compiler" (Dunn, column 3, line 52). In contrast, in the approach of this invention, program control is shifted through compensation codes to the exception handler in order to "absorb the difference between the point of origin of the execution occurrence points and the exception handler." For example, the present invention preferably performs the algorithms shown in Figures 4 to 7 in order to compensate a register image between the point of origin of the exception occurrence points and the exception handler. To take an example shown in Figure 10 of this application, Dunn's approach does not generate "copy i and j variable values to R1 and R2" in the recovery block.

Another important difference between Dunn, et al. and the preferred embodiment of this invention is that Dunn, et al. considers only hardware trapping instructions, which directly jump to the operating system when detecting an exception, as shown in Dunn, et al, Figures 2 and 5. In contrast, the preferred embodiment of the present invention may be used to cover not only

hardware trapping instructions but also software exception checking instructions, such as SIZECHECK. For software checking, even when detecting an exception, execution is not transferred to the operating system but is directly transferred to the corresponding exception handler, as shown in Figure 1 of this application. Therefore, the Dunn, et al approach cannot, for example, transform Figure 16 of this application into Figure 18 regarding SIZECHECK.

Each of independent Claims 1, 6, 7, 8, 12, 13, 15 and 16 describe the above-discussed feature of this invention relating to the way the exception handler is used. More specifically, Claims 1 and 6 describe a program modification unit for modifying the object program in order to absorb a difference in content between the point of origin of an exception process, which occurs in response to the execution of a command in said object program, and an exception handler whereat said exception process is performed. Claim 7 describes the step of generating in a basic block compensation code when a difference exists between the point of origin of said exception process and an exception handler whereat said exception process is performed, for compensating for said difference. Claim 8, similarly, describes the step of generating in a basic block when a difference exists, a compensation code for compensating for said difference, and a code for, after said compensation code has been obtained, moving program control to an exception handler whereat said exception process is performed.

Claims 12, 15 and 16 describe a process for, when a difference in content exists between the point of origin of an exception process and an exception handler whereat said exception process is performed, generating in said basic block compensation code for compensating for said difference. Analogously, Claim 13 describes a function for, when an exception process has occurred, shifting program control to an exception handle whereat said exception process is run,

and for, when a difference in content exists between the point of origin of said exception process and said exception handler whereat said exception process is run, compensating for said difference before program control is shifted to said exception handler.

The other references of record have been considered, and these other references, whether considered individually or in combination also do not disclose or suggest the use of the exception handler in this way. For example, Ghosh was cited for its disclosure of try-catch block for handling exceptions, but this reference does not use an exception handler to identify a point in a process with respect to which differences should be measured or compensated for, as described in Claims 1, 6, 7, 8, 12, 13, 15 and 16.

Another important feature disclosed in this application that is not disclosed in or suggested by Dunn, et al. involves the division of the software code relating to the exception process. More specifically, in accordance with this feature, an application program (or software) detects and throws an exception; however, the Dunn, et, al. approach (Dunn, column 6, lines 1-35) supposes that the hardware detects and throws an exception. The approach of the present invention divides a software exception check into a detection portion and a throwing exception portion within "the application program," as shown in Figure 26 of this application. Therefore, this approach of this invention enables that an execution is directly transferred within the application program. In contrast, the Dunn, et al. technique is limited by the fact that an execution must be transferred via the operating system (Dunn, et al. column 6, lines 1-35).

Independent Claim 10 describes this feature. More particularly, Claim 10 is directed to a method for optimizing a program to increase processing efficiency, and this claim positively sets forth the step of dividing software code, in an object program, that may cause an exception process into software code for determining whether an exception process has occurred and software code for actually causing an exception process. Claim 10 goes on to describe that a control flow graph is designed so that when an exception process occurs, program control is shifted to the code that actually caused the exception process.

The other references of record also fail to disclose or suggest this feature of Claim 10. In particular, again, Ghosh was cited for its teaching of using try-catch blocks for handling exceptions, but this reference does not suggest the above-described procedure for dividing a software exception check into a detection portion and a throwing exception portion.

Because of the above-discussed differences between Claims 1, 6, 7, 8, 10, 12, 13, 15 and 16 and the prior art and because of the advantages associated with those differences, it cannot be said that any of these claims is anticipated by or obvious in view of the prior art. Accordingly, these claims patentably distinguish over the prior art and are allowable. Claims

Claims 2-5, 17 and 18 are dependent from Claim 1 and are allowable therewith; and Claim 9 is dependent from, and is allowable with, Claim 8. Likewise, Claims 11 and 14 are dependent from and are allowable with Claims 10 and 13 respectively. Consequently, the Examiner is requested to reconsider and to withdraw the rejection of Claims 1, 2, 4, 6, 7 and 10-16 under 35 U.S.C. §102 and the rejection of Claims 3, 5, 8 and 9 under 35 U.S.C. §103, and to allow Claims 1-18.

For the reasons set forth above, the Examiner is respectfully asked to reconsider and to withdraw the objections to the specification and to Claim 5 and the rejection of Claim 5 under 35 U.S.C. §112. The Examiner is also requested to reconsider and to withdraw the rejection of Claims 1, 2, 4, 6, 7, and 10-16 under 35 U.S.C. §102 and the rejection of Claims 3, 5, 8 and 9 under 35 U.S.C. §103, and to allow Claims 1-18. If the Examiner believes that a telephone conference with Applicants' Attorneys would be advantageous to the disposition of this case, the Examiner is asked to telephone the undersigned.

Respectfully submitted,

John S. Sensny
Registration No. 28,757
Attorney for Applicants

Scully, Scott, Murphy & Presser
400 Garden City Plaza
Garden City, New York 11530
(516) 7472-4343

JSS:jy